1 Assignment 11 (100 points)

To submit your work, collect your inputs not provided as part of the assignment, the required outputs, and your answers to the written questions in pdf format (preferred) or Word, or .txt . Clearly label all of your work. For the math-oriented portion of the assignment, you may typeset in LaTeX or Word or provide a clear photograph of a hand-written solution. Work that is not legible will not be graded. Name a zip file `CS585_Assignment11_username.zip` and submit with web-submit. Please facilitate grading by labeling the code that you change with `//modified by deht`

1.1 Learning Objectives

- Understand what data association is and why it is necessary during multi-target tracking
- Understand some of the issues that arise when performing multi-target tracking
- See one way a Kalman filter can be used for data association during multi-target tracking

1.2 Programming Assignment

For this assignment, you are given a complete, working program that will track multiple red objects, but you will need to make some modifications. When you run the program, press 't' to start the object detection and tracking. Use the sliders to adjust the red threshold and minimum area thresholds, if necessary. The detection output is shown in the "Just Red" window. Once you have good thresholds in place, I recommend stopping the tracking (by pressing 't' again) and then restarting. As you move the red object(s), you will see colored tails depicting the paths of the objects over time. To help distinguish track identity, the tracks are colored blue, green, or yellow.

In Assignment 3, where we used a red object to draw a frame that was put around our faces, we had one global variable that represented the entire history of the track. Now, since we have multiple objects, we need to maintain track history for each object. To facilitate this, I have provided a "TrackedObject" class that contains the history of the track, as well as a Kalman filter. The Kalman filter is used in two ways in this context. First, as we find new measurements (object detections), we estimate the position of the track and store the state estimate rather than the raw detection location. Second, and slightly more importantly, we also use it to predict the future state of the track, so that we can compare the locations of our new object detections to the predicted locations of our tracks.

The data association implemented in `trackRedObjects` is the simplest data association you could imagine. For each track, it computes the distance between the predicted location of the track and the location of each detection. Each track then uses whichever detection was closest to its predicted location, *even if another track has already used that detection.* As you can imagine, this can lead to some pretty bad behavior. You will need to fix it.

Since we are only considering two frames, you only need to do bipartite matching. Your program will only need to work for up to three objects. I recommend that you enumerate all possible associations (there will be 8 if you have three objects and three detections), compute the total cost of each association, and then choose the association with the lowest cost. I think that the Hungarian algorithm is overkill for this very restricted application.

In the given program, tracks are only initiated when there are no tracks already present. This seems undesirable, since you need to have both objects shown to the camera when you press 't' or it will not track both objects. You will need to augment your data association and track management to use unclaimed detections to start new tracks. In the given implementation, tracks are only terminated when you press 't'. You will need to augment your data assocation and track management to terminate tracks that do not have an associated measurement.

To recap, the requirements are:

1. In the `TrackedObject` constructor, it sets up a Kalman filter state transition matrix. You must write down what the matrix is that it constructs, and explain what it means.

2. Spend a little time trying to track more than one object with the given implementation. Document three undesirable behaviors and speculate about why they arise. One or two sentences per idea is sufficient. Hint: one undesirable behavior to look out for is a "track switch" behavior which will most likely occur when your objects touch or occlude each other. You may elaborate on some of the issues I outlined in the description above.

3. Fix the data association in `trackRedObjects`. Make whatever changes to the code you need. To start, your implementation can work with only two objects. Submit a sequence of five (possibly non-consecutive) images showing your program tracking through a situation that would cause a track switch with the original implementation.

4. Enable track initiation. If there are more detections than tracks at a particular frame, the unclaimed detection should be used to start a new track. Submit a sequence of five (possibly non-consecutive) images showing a situation where there is a track present in the image, and then a new object is introduced, resulting in a new track

5. Enable track termination. If there are fewer detections than tracks, the track that does not receive a detection should be terminated. Submit an image sequence of five (possibly non-consecutive) images showing two tracked objects, and then one of the tracks disappearing as the object is moved out of the field of view.

6. Extend your data association to work with up to three tracks and detections. Submit an image sequence of five (possibly non-consecutive) images showing your program tracking 3 objects.

1.3 Lecture Preparation

The last topic we will cover this semester will be the Fourier transform. This resource is very helpful: `http://www.thefouriertransform.com/#introduction`. Read through the introduction and watch the 9 minute video lecture.

1. What is the basic idea of the Fourier transform?